APPENDIX

List of Contents

Attachment 1

```
;*****************************************************************
      TITLE "autonomous sine wave tone generator"
        LIST P=16C621A, R=DEC

        INCLUDE   <P16C621A.INC>
      __CONFIG    _BODEN_OFF&_CP_OFF&_PWRTE_ON&_WDT_OFF&_HS_OSC
;
;*****************************************************************
;       File:           TONE.ASM
;     Author:           Jeremy Sommer
;       Date:           07/05/00
;     Assembler:  MPASM V01.40
;       Xtal:           4.14 Mhz
;       Inst Clk:       1.035 Mhz (966.2 nSec)
;*****************************************************************
;       Description:
;
;     ROM Usage:  words
;
;     RAM Usage:  bytes
;
;************************* Constant Definition   **********************

STEP#       EQU   .8          ; Number of steps

;************************  Register Definition  **********************

COMP_FLAG   EQU   0x20        ; COMPARATOR OUTPUTS
REP_ID            EQU   0x21        ; REPEATER ID (0x00 = CPE end)

;************************  Bit Definition  ***************************

SIGCOMP_OUT EQU   .6          ; Signal comparator output
VCOMP_OUT   EQU   .6          ; Supply Voltage comparator output
IZCOMP_OUT  EQU   .7          ; Shunt Current comparator output
TEMPCOMP_OUT      EQU   .7          ; Temperature comparator output
RB1         EQU   .1          ; PORT_B<1>

;*****************************************************************
;                       Reset Vector
;*****************************************************************

        org     0x000
        goto    Start         ; Begining of Program


;*****************************************************************
;                            Main Routine
;*****************************************************************
Start
      bcf   STATUS,RP0  ; Select bank 0
      clrf  PORTA       ; Initialize Port_A by setting output latches
      clrf  PORTB       ; Initialize Port_B by setting output latches

      bsf   STATUS,RP0  ; Select register bank 1
      movlw 0xDF        ; Configure TMR0 to run off Fclk/4 (for debugging)
      movwf OPTION_REG  ;
```

```
      movlw 0x88          ; Configure for internal voltage reference of 2.5V nom.
      movwf VRCON         ;
      movlw 0x1F          ; Set Port_A comparator as inputs
      movwf TRISA         ;
      movlw 0xFF          ; Set Port_B as inputs
      movwf TRISB         ;

      bcf   STATUS,RP0    ; Select register bank 0
      movf  PORTB,0             ; Identify repeater {0xFF = CO end of 2, 0x00 =
CPE end}
      movwf REP_ID              ;
      movlw       0x02          ; Configure for 4 inputs muxed to 2 comparators
      movwf       CMCON         ; Enable pins for I/O functions
      movlw 0xFF          ; Set Port_B to 0xFF (.255)
      movwf PORTB         ;

      bsf   STATUS,RP0    ; Select register bank 1
      movlw 0x00          ; Set Port_B as outputs
      movwf       TRISB           ;

      bcf     STATUS,RP0        ; Select register bank 0 (Initialization complete)

HealthCheck
                          ; (Placeholder for health verification via comparators)

SelectTone
      bcf   CMCON,CIS     ; Select Signal and Supply Voltage comparators
      btfss REP_ID,RB1    ; If REP_ID<1> is 0 (CPE end repeater),
      goto  Tone5_Loop    ;   test for tone5 generation
;     goto  Tone4_Loop    ; else test for tone4 generation
                          ; (commented out due to sequentiality)

Tone4_Loop
      btfsc CMCON,SIGCOMP_OUT ; If Signal comparator is high (low signal power),
      goto  Tone4_Loop    ;   stay static (no tone)

tone4                     ; 17.25 kHz sine at 2/60 UI intervals;  4 instructions
overhead
      movlw   .247              ; 4
      movwf PORTB         ;
      movlw   .233              ; 6
      movwf PORTB         ;
      movlw   .215              ; 8
      movwf PORTB         ;
      movlw   .193              ; 10
      movwf PORTB         ;
      movlw   .168              ; 12
      movwf PORTB         ;
      movlw   .141              ; 14
      movwf PORTB         ;
      movlw   .114              ; 16
      movwf PORTB         ;
      movlw   .87         ; 18
      movwf PORTB         ;
      movlw   .62         ; 20
      movwf PORTB         ;
      movlw   .40         ; 22
```

```
        movwf PORTB         ;
        movlw   .22         ; 24
        movwf PORTB         ;
        movlw   .8          ; 26
        movwf PORTB         ;
        movlw   .0          ; 28
        movwf PORTB         ;
        movlw   .0          ; 30
        movwf PORTB         ;
        movlw   .0          ; 32
        movwf PORTB         ;
        movlw   .8          ; 34
        movwf PORTB         ;
        movlw   .22         ; 36
        movwf PORTB         ;
        movlw   .40         ; 38
        movwf PORTB         ;
        movlw   .62         ; 40
        movwf PORTB         ;
        movlw .87           ; 42
        movwf PORTB         ;
        movlw   .114              ; 44
        movwf PORTB         ;
        movlw   .141              ; 46
        movwf PORTB         ;
        movlw   .168              ; 48
        movwf PORTB         ;
        movlw   .193              ; 50
        movwf PORTB         ;
        movlw .215          ; 52
        movwf PORTB         ;
        movlw .233          ; 54
        movwf PORTB         ;
        movlw .247          ; 56
        movwf PORTB         ;
        movlw .255          ; 58
        movwf PORTB         ;

        goto  Tone4_Loop    ; Go back to recheck signal level

Tone5_Loop
        btfsc CMCON,SIGCOMP_OUT ; If Signal comparator is high (low signal power),
        goto  Tone5_Loop    ;   stay static (no tone)

tone5                       ; 21.5625 kHz sine at 2/48 UI intervals;  4 instructions
overhead

        movlw   .242              ; 4
        movwf PORTB         ;
        movlw   .221              ; 6
        movwf PORTB         ;
        movlw   .193              ; 8
        movwf PORTB         ;
        movlw   .162              ; 10
        movwf PORTB         ;
        movlw   .127              ; 12
        movwf PORTB         ;
```

```
movlw   .93         ; 14
movwf PORTB         ;
movlw   .62         ; 16
movwf PORTB         ;
movlw   .34         ; 18
movwf PORTB         ;
movlw   .13         ; 20
movwf PORTB         ;
movlw   .0          ; 22
movwf PORTB         ;
movlw   .0          ; 24
movwf PORTB         ;
movlw   .0          ; 26
movwf PORTB         ;
movlw   .13         ; 28
movwf PORTB         ;
movlw   .34         ; 30
movwf PORTB         ;
movlw   .62         ; 32
movwf PORTB         ;
movlw   .93         ; 34
movwf PORTB         ;
movlw   .128              ; 36
movwf PORTB         ;
movlw   .162              ; 38
movwf PORTB         ;
movlw   .193              ; 40
movwf PORTB         ;
movlw .221          ; 42
movwf PORTB         ;
movlw   .242              ; 44
movwf PORTB         ;
movlw   .255              ; 46
movwf PORTB         ;

goto  Tone5_Loop  ; Go back to recheck signal level


  END               ; That's all Folks !
```

Attachment 2

```
;*****************************************************************
      TITLE "non-autonomous sine wave tone generator"
        LIST P=16C621A, R=DEC

        INCLUDE   <P16C621A.INC>
      __CONFIG    _BODEN_ON&_CP_OFF&_PWRTE_ON&_WDT_OFF&_HS_OSC
;
;*****************************************************************
;       File:           TONE.ASM
;     Author:           Jeremy Sommer
;       Date:           07/05/00
;     Assembler:  MPASM V01.40
;       Xtal:           16.56 Mhz
;       Inst Clk:       4.14 Mhz (241.5 nSec)
;*****************************************************************
;       Description:
;
;     ROM Usage:  words
;
;     RAM Usage:  bytes
;
;************************ Constant Definition   *********************

REP_IDMASK# EQU   0xC0        ; Mask of CMD and PORT_B bits which select
                              ; repeater ID
CO_ID#            EQU   0x40        ; REP_ID for CO end of multiple repeaters
CPE_ID#           EQU   0xC0        ; REP_ID for CPE end
VRMASK#           EQU   0x2F        ; Mask of valid VAL bits to use for Vref
CMODE#            EQU   0x02        ; Comparator mode for all four comparators
COUTMASK#   EQU   0xC0        ; Comparator output bits
HI_TEMP#    EQU   0x8D        ; High temperature health threshold -> 82C
HI_SUPPLY#  EQU   0x8C        ; High supply voltage health threshold -> 34.4V
LO_SUPPLY#  EQU   0x87        ; Low supply voltage health threshold -> 25.8V
HI_IZ#            EQU   0x8D        ; High shunt current health threshold ->
32.8mA
LO_IZ#            EQU   0xA1        ; Low shunt current health threshold ->
2.1mA
PRESCALE#   EQU   .0;   .7          ; Prescale 2^(7+1)=256
INIT_TMSB#  EQU   .3;   .15         ; Initial TIME_MSB
NSAMPLES#   EQU   .4;   .127        ; Number of samples per burst
NMATCHING#  EQU   .3;   .102        ; Minimum number of matching samples
                              ; (should be set to 80% of NSAMPLES#)
ADDRN_A#    EQU   0x3A        ; Latest Phase A sample register
ADDRN_B#    EQU   0x4A        ; Latest Phase B sample register
ADDRN#            EQU   0x5A        ; Latest temporary sample register
NUM_ADDRS#  EQU   0x0A        ; Number of sample registers per phase

                              ; (from first bits received)
PAT12#            EQU   0x54        ; Pattern1 MSBs minus 1
PAT11#            EQU   0xAB        ; Pattern1 ...  minus 1
PAT10#            EQU   0x54        ; Pattern1 LSBs minus 1
PAT22#            EQU   0xA9        ; Pattern2 MSBs minus 1
PAT21#            EQU   0x52        ; Pattern2 ...  minus 1
PAT20#            EQU   0xA9        ; Pattern2 LSBs minus 1
                              ; (to last bits received)

;************************  Register Definition  ********************
```

```
PHASE        EQU    0x20        ; PHASE REGISTER (0x00=A, 0xFF=B)
TEMP1        EQU    0x21        ; TEMPORARY REGISTER 1
TEMP2        EQU    0x22        ; TEMPORARY REGISTER 2
TEMP3        EQU    0x23        ; TEMPORARY REGISTER 3
W_TEMP              EQU    0x24        ; TEMPORARY W REGISTER (both banks)
STATUS_TEMP  EQU    0x25        ; TEMPORARY STATUS REGISTER
TIME_MSB     EQU    0x26        ; TMR0 OVERFLOW COUNTER
SAMPLE              EQU    0x27        ; SAMPLE VALUE (0 or 1)
COUNT        EQU    0x28        ; COUNT OF SAMPLES TO GO
COUNT1              EQU    0x29        ; COUNT OF SAMPLES=1
REP_ID              EQU    0x2A        ; REPEATER ID (0x00 = CPE end)
TONE_ID             EQU    0x2B        ; TONE IDENTIFICATION REGISTER (0x00 =
tone10)
CMD          EQU    0x2C        ; COMMAND (CONCATENATED)
VAL          EQU    0x2D        ; VALUE (CONCATENATED)
PAT12_A             EQU    0x31        ; PATTERN12 (PHASE A)
PAT11_A             EQU    0x32        ; PATTERN11 (PHASE A)
PAT10_A             EQU    0x33        ; PATTERN10 (PHASE A)
CMD1_A              EQU    0x34        ; COMMAND1 (PHASE A)
CMD0_A              EQU    0x35        ; COMMAND0 (PHASE A)
VAL1_A              EQU    0x36        ; VALUE1 (PHASE A)
VAL0_A              EQU    0x37        ; VALUE0 (PHASE A)
PAT22_A             EQU    0x38        ; PATTERN22 (PHASE A)
PAT21_A             EQU    0x39        ; PATTERN21 (PHASE A)
PAT20_A             EQU    0x3A        ; PATTERN20 (PHASE A)
PAT12_B             EQU    0x41        ; PATTERN12 (PHASE B)
PAT11_B             EQU    0x42        ; PATTERN11 (PHASE B)
PAT10_B             EQU    0x43        ; PATTERN10 (PHASE B)
CMD1_B              EQU    0x44        ; COMMAND1 (PHASE B)
CMD0_B              EQU    0x45        ; COMMAND0 (PHASE B)
VAL1_B              EQU    0x46        ; VALUE1 (PHASE B)
VAL_B        EQU    0x47        ; VALUE0 (PHASE B)
PAT22_B             EQU    0x48        ; PATTERN22 (PHASE B)
PAT21_B             EQU    0x49        ; PATTERN21 (PHASE B)
PAT20_B             EQU    0x4A        ; PATTERN20 (PHASE B)
PAT12        EQU    0x51        ; PATTERN12
PAT11        EQU    0x52        ; PATTERN11
PAT10        EQU    0x53        ; PATTERN10
CMD1         EQU    0x54        ; COMMAND1
CMD0         EQU    0x55        ; COMMAND0
VAL1         EQU    0x56        ; VALUE1
VAL0         EQU    0x57        ; VALUE0
PAT22        EQU    0x58        ; PATTERN22
PAT21        EQU    0x59        ; PATTERN21
PAT20        EQU    0x5A        ; PATTERN20

;************************  Bit Definition  ************************

VREN         EQU    .7          ; Vref enable bit
SIGCOMP_OUT  EQU    .6          ; Signal comparator output
VCOMP_OUT    EQU    .7          ; Supply Voltage comparator output
IZCOMP_OUT   EQU    .7          ; Shunt Current comparator output
TEMPCOMP_OUT        EQU    .6          ; Temperature comparator output
CIS          EQU    .3          ; Comparator input switch
                                ; (=0 for Signal comparator
                                ;     and Temperature comparator,
```

```
                                ;  =1 for Supply Voltage comparator
                                ;     and Shunt Current comparator)
RA4             EQU    .4       ; PORT_A<4>
RB1             EQU    .1       ; PORT_B<1>
INTF            EQU    .1       ; INTCON<1>
T0IF            EQU    .2       ; INTCON<2>
RA0             EQU    .0       ; PORT_A<0>
POR             EQU    .1       ; PCON<POR>
BO              EQU    .0       ; PCON<BO>

;*****************************************************************
;                        Reset Vector
;*****************************************************************

        org     0x000
        goto    Start           ; Begining of Program

      nop               ; PC=1
      nop               ; PC=2
      nop               ; PC=3

;*****************************************************************
;                            Interrupt Service Routine
;*****************************************************************

ISR
                        ; breakpoint
      movwf W_TEMP              ; copy W to temp register,
                        ; could be in either bank
      swapf STATUS,0    ; swap status to be saved into W
      bcf   STATUS,RP0  ; change to bank 0 regardless
                        ; of current bank
      movwf STATUS_TEMP ; save status to bank 0 register

      btfsc INTCON,INTF ; If interrupt is external INT
      goto  WaitForLoopback   ;   then goto WaitForLoopback
      btfss INTCON,T0IF ; else if interrupt is not TMR0
      goto  BogusInterrupt    ;   then the interrupt was bogus
      call  DecTimeMSB  ; else decrement TIME_MSB
      goto  DoRetfie    ; return from ISR

BogusInterrupt                  ; bogus interrupt!!!!!!!
      movlw 0xB0        ; Clear interrupt flags and make sure that
                        ;   no unused interrupts are enabled somehow
      andwf INTCON,1    ;
DoRetfie
;     bcf   STATUS,RP0  ; change to bank 0 in case ISR changed bank
      swapf STATUS_TEMP,0       ; swap STATUS_TEMP register into W,
                        ; sets bank to original state
      movwf STATUS              ; move W into STATUS register
      swapf W_TEMP,1    ; swap W_TEMP
      swapf W_TEMP,0    ; swap W_TEMP into W
      retfie                    ; Return from ISR

;*****************************************************************
;                            Main Routine
;*****************************************************************
```

```
Start
      bcf   STATUS,RP0  ; Select bank 0
      clrf  PORTA       ; Initialize Port_A by setting output latches
      clrf  PORTB       ; Initialize Port_B by setting output latches

      bsf   STATUS,RP0  ; Select register bank 1
      movlw 0xD0        ; Configure TMR0 to run off Fclk/4, with prescale
      iorlw PRESCALE#   ;
      movwf OPTION_REG  ;
      movlw 0x0F        ; Set Port_A comparator as inputs, except RA4 as output
      movwf TRISA       ;
      movlw 0xFF        ; Set Port_B as inputs
      movwf TRISB       ;

      bcf   STATUS,RP0  ; Select register bank 0
      movlw CO_ID#            ; Prepare to set REP_ID to CO_ID#
      btfss PORTB,7           ; If PORT_B<7> = 0 (indicates the CPE end
repeater)
      movlw CPE_ID#           ; Prepare to set REP_ID to CPE_ID#
      movwf REP_ID            ; Set REP_ID
      movlw       0x02        ; Configure for 4 inputs muxed to 2 comparators
      movwf       CMCON       ; Enable pins for I/O functions
      movlw 0xFF        ; Set Port_B to 0xFF (.255)
      movwf PORTB       ;

      bsf   STATUS,RP0  ; Select register bank 1
      movlw 0x01        ; Set Port_B as outputs except for RB0, which is the INT
input
      movwf       TRISB           ;

      bcf   STATUS,RP0  ; Select register bank 0 (Initialization complete)
      goto  TestHealth  ; Verify card health, generate ACK_tone if healthy

DecTimeMSB
      bcf   INTCON,T0IF ; Clear the T0IF interrupt
      bsf   INTCON,GIE  ; Set Global interrupt enable since ISR may have been
                        ; escaped without return
      decfsz      TIME_MSB,1  ; Decrement TIME_MSB and if it is still positive
      return                  ;   return from DecTimeMSB                ;
      movlw INIT_TMSB#  ; else reinitialize TIME_MSB register,
      movwf TIME_MSB    ;
;     goto  TakeSample  ; and then sample the output of the Signal Comparator
                        ; (commented out due to sequentiality)

TakeSample              ; TAKE SAMPLE BURST,
                        ; VALIDATE CORRELATION,
                        ; SHIFT INTO SAMPLE REGISTER FOR CURRENT PHASE,
                        ; TRANSFER TO TEMPORARY SAMPLE REGISTER
      comf  PHASE,1           ; switch phase of sampling (alternate A and B)
      call  GetSamples  ; accumulate NSAMPLES# samples
      movwf TEMP1       ; recover sample value (0=0,1=1,.255=invalid)
      btfsc TEMP1,7           ; If invalid sample,
      goto  ResetCurrentPhase ;   clear sample registers of current phase
                        ;   and start waiting all over again
      movlw ADDRN_A#    ; Starting with latest Phase A register
      btfsc PHASE,0           ; If PHASE = 1 (B)
      movlw ADDRN_B#    ;   switch to latest Phase B register
```

```
        movwf FSR           ; Initialize FSR to latest sample register
        rrf   TEMP1,1           ; Put sample value (0=0,1=1) in carry bit
        movlw NUM_ADDRS#    ; Put number of sample addresses per phase
        movwf TEMP1         ;   into the TEMP1 register
RotNextReg
        rlf   INDF,1            ; Rotate register left through carry bit,
                            ; note that the first time this puts the
                            ; latest sample into bit 0 of latest sample
                            ; register
        decf  FSR,1         ; Move to next earliest sample register
        decfsz      TEMP1,1             ; If not all registers are done,
        goto  RotNextReg    ;   then go back for next register rotation

                            ; Now we're ready to check for loopback!
                            ; First we need to copy all the current phase
                            ; registers into temporary sample registers.
        movlw ADDRN#            ; Start with the latest temporary sample register
        movwf TEMP2         ; Store its address in TEMP2
        movlw ADDRN_A#      ; Starting with latest Phase A register
        btfsc PHASE,0           ; If PHASE = 1 (B)
        movlw ADDRN_B#      ;   switch to latest Phase B register
        subwf TEMP2,1           ; Subtract phase register address from TEMP2 and
                            ;   store the result in TEMP2
        movwf FSR           ; Initialize FSR to latest phase sample register
        movlw NUM_ADDRS#    ; Put number of sample addresses per phase
        movwf TEMP1         ;   into the TEMP1 register
XfrNextReg
        movf  INDF,0            ; Get the phase register contents
        movwf TEMP3         ; Transfer the phase register contents to TEMP3
        movf  TEMP2,0           ; Get the address offset from TEMP2
        addwf FSR,1         ; Point to the temporary sample register
                            ;   by adding the address offset
        movf  TEMP3,0           ; Get the phase register contents
        movwf INDF          ; Transfer the phase register contents to the
                            ; temporary sample register
        decf  FSR,1         ; Move to next earliest temporary sample register
        movf  TEMP2,0           ; Get the address offset from TEMP2
        subwf FSR,1         ; Point to the next earliest phase register
                            ;   by subtracting the address offset
        decfsz      TEMP1,1             ; If not all registers are done,
        goto  XfrNextReg    ;   then go back for next register transfer
                            ; Else the transfer is over, and it's time
                            ; to check the patterns!

CheckPattern                    ; CHECK PATTERN BITS FOR VALID MATCH
        movlw PAT12#            ; Put PAT12# into W register
        subwf PAT12,1       ; Subtract PAT12# from PAT12
        decfsz      PAT12,1     ; If pattern did not match
        goto  InfiniteLoop      ;   wait indefinitely
        movlw PAT11#            ; Put PAT11# into W register
        subwf PAT11,1       ; Subtract PAT11# from PAT11
        decfsz      PAT11,1     ; If pattern did not match
        goto  InfiniteLoop      ;   wait indefinitely
        movlw PAT10#            ; Put PAT10# into W register
        subwf PAT10,1       ; Subtract PAT10# from PAT10
        decfsz      PAT10,1     ; If pattern did not match
        goto  InfiniteLoop      ;   wait indefinitely
```

```
        movlw PAT22#            ; Put PAT22# into W register
        subwf PAT22,1     ; Subtract PAT22# from PAT22
        decfsz      PAT22,1     ; If pattern did not match
        goto  InfiniteLoop      ;   wait indefinitely
        movlw PAT21#            ; Put PAT21# into W register
        subwf PAT21,1     ; Subtract PAT21# from PAT21
        decfsz      PAT21,1     ; If pattern did not match
        goto  InfiniteLoop      ;   wait indefinitely
        movlw PAT20#            ; Put PAT20# into W register
        subwf PAT20,1     ; Subtract PAT20# from PAT20
        decfsz      PAT20,1     ; If pattern did not match
        goto  InfiniteLoop      ;   wait indefinitely
                          ; Pattern matches!

CheckOverallParity              ; CHECK CMD AND VAL BITS FOR PARITY
        movf  CMD1,0      ; Put CMD1 into W register
        call  CheckParity ; Check that bits 7,6 and 3,2
                          ; are complements of bits 5,4
                          ; and 1,0 respectively
        movwf TEMP1       ; Recover return value
        btfsc TEMP1,7           ; If return value indicates invalid parity
        goto  InfiniteLoop      ;   wait indefinitely
        movf  CMD0,0      ; Put CMD0 into W register
        call  CheckParity ; Check that bits 7,6 and 3,2
                          ; are complements of bits 5,4
                          ; and 1,0 respectively
        movwf TEMP1       ; Recover return value
        btfsc TEMP1,7           ; If return value indicates invalid parity
        goto  InfiniteLoop      ;   wait indefinitely
        movf  VAL1,0      ; Put VAL1 into W register
        call  CheckParity ; Check that bits 7,6 and 3,2
                          ; are complements of bits 5,4
                          ; and 1,0 respectively
        movwf TEMP1       ; Recover return value
        btfsc TEMP1,7           ; If return value indicates invalid parity
        goto  InfiniteLoop      ;   wait indefinitely
        movf  VAL0,0      ; Put VAL0 into W register
        call  CheckParity ; Check that bits 7,6 and 3,2
                          ; are complements of bits 5,4
                          ; and 1,0 respectively
        movwf TEMP1       ; Recover return value
        btfsc TEMP1,7           ; If return value indicates invalid parity
        goto  InfiniteLoop      ;   wait indefinitely
                          ; Parity is OK!

InterpretCmdVal                 ; INTERPRET CMD AND VAL BITS FOR ACTION
        clrf  TEMP2       ; Clear TEMP2
        movf  VAL1,0            ; Put VAL1
        movwf TEMP1       ; into TEMP1
        call  ConcatenateBits   ; Shift the concatenated command bits
                          ; into TEMP2
        movf  VAL0,0            ; Put VAL0
        movwf TEMP1       ; into TEMP1
        call  ConcatenateBits   ; Shift the concatenated command bits
                          ; into TEMP2
        movf  TEMP2,0           ; Put TEMP2
        movwf VAL         ; into VAL
```

```
      clrf  TEMP2        ; Clear TEMP2
      movf  CMD1,0             ; Put CMD1
      movwf TEMP1        ; into TEMP1
      call  ConcatenateBits   ; Shift the concatenated command bits
                         ; into TEMP2
      movf  CMD0,0             ; Put CMD0
      movwf TEMP1        ; into TEMP1
      call  ConcatenateBits   ; Shift the concatenated command bits
                         ; into TEMP2
      movf  TEMP2,0            ; Put TEMP2
      movwf CMD          ; into CMD

      andlw REP_IDMASK# ; Select only the REP_ID bits of CMD,
      movwf TEMP1        ; and put them in TEMP1
      incf  TEMP1,1            ; If TEMP1 = 0 (indicating CMD is for all
repeaters)
      decfsz      TEMP1,1           ;
      goto  CheckREP_ID ; Check whether this is repeater being addressed
      goto  CheckCmd    ;
CheckREP_ID
      movf  REP_ID,0    ; Put REP_ID in W register
      subwf TEMP1,1            ; Subtract REP_ID from CMD REP_ID
      incf  TEMP1,1            ; If the REP_ID's don't match
      decfsz      TEMP1,1           ;
      goto  WaitForLoopback   ;   then clear all sample registers and start
                         ;   waiting all over again
;     goto  CheckCmd    ; else proceed with checking the command
                         ; (commented out due to sequentiality)
CheckCmd
      decfsz      CMD,1       ; If command was not 1,
      goto  CheckFor2   ;   check for 2
      goto  ACK_now            ; else force an ACK
CheckFor2
      decfsz      CMD,1       ; If command was not 2,
      goto  CheckFor3   ;   check for 3
      goto  TestHealth  ; else test if health check = OK
CheckFor3
      decfsz      CMD,1       ; If command was not 3,
      goto  CheckFor4   ;   check for 4
      goto  TestSupply  ; else test supply voltage
CheckFor4
      decfsz      CMD,1       ; If command was not 4,
      goto  CheckFor5   ;   check for 5
      goto  TestIZ             ; else test shunt current
CheckFor5
      decfsz      CMD,1       ; If command was not 5,
      goto  CheckFor6   ;   check for 6
      goto  TestTemp    ; else test temperature
CheckFor6
      decfsz      CMD,1       ; If command was not 6,
      goto  CheckFor7   ;   check for 7
      goto  TestReset   ; else test if power-on reset occurred since last
                         ;   check via this command
CheckFor7
      decfsz      CMD,1       ; If command was not 7,
      goto  WaitForLoopback   ;   clear all sample registers and start
                         ;   waiting all over again
```

```
;       goto  TestBrownout      ; test if brownout reset occurred since last
                              ;   check via this command
                              ; (commented out due to sequentiality)

TestBrownout
        bsf   STATUS,RP0  ; Select bank 1
        movf  PCON,0            ; Put PCON into W register
        movwf TEMP1       ;   and then into TEMP1
        bcf   STATUS,RP0  ; Select bank 0
        btfsc TEMP1,BO    ; If a brownout has not occurred,
        goto  WaitForLoopback   ;   clear all sample registers and start
                              ;   waiting all over again
        bsf   STATUS,RP0  ; else select bank 1
        bsf   PCON,BO           ;   and set the brownout detect flag (normal)
        bcf   STATUS,RP0  ; Select bank 0
        goto  ACK_now           ; and generate an ACK tone

TestReset
        bsf   STATUS,RP0  ; Select bank 1
        movf  PCON,0            ; Put PCON into W register
        movwf TEMP1       ;   and then into TEMP1
        bcf   STATUS,RP0  ; Select bank 0
        btfsc TEMP1,POR   ; If a power-on reset has not occurred,
        goto  WaitForLoopback   ;   clear all sample registers and start
                              ;   waiting all over again
        bsf   STATUS,RP0  ; else select bank 1
        bsf   PCON,POR    ;   and set the power-on reset detect flag (normal)
        bcf   STATUS,RP0  ; Select bank 0
        goto  ACK_now           ; and generate an ACK tone

TestSupply
        call  PresetForSupply   ; Preset TEMP1 for Supply Test
        call  DoCompCheck ; Do the comparator check
        goto  EvaluateAnswer    ; Act on the result

TestIZ
        call  PresetForIZ ; Preset TEMP1 for Shunt Current Test
        call  DoCompCheck ; Do the comparator check
        goto  EvaluateAnswer    ; Act on the result

TestTemp
        call  PresetForTemp     ; Preset TEMP1 for Temperature Test
        call  DoCompCheck ; Do the comparator check
        goto  EvaluateAnswer    ; Act on the result

TestHealth
        movlw HI_TEMP#    ; set VAL corresponding to T=82 deg. C
        movwf VAL         ;
        call  PresetForTemp     ;
        call  DoCompCheck ;
        movwf TEMP1       ; Retrieve answer (0x00=yes, 0xFF=no)
        btfss TEMP1,7           ; If yes (Temp too high),
        goto  WaitForLoopback   ;   clear all sample registers and start
                              ;   waiting all over again
        movlw HI_SUPPLY#  ; else set VAL corresponding to V=34.4V
        movwf VAL         ;
        call  PresetForSupply   ;
```

```
        call  DoCompCheck ;
        movwf TEMP1       ; Retrieve answer (0x00=yes, 0xFF=no)
        btfss TEMP1,7           ; If yes (Supply Voltage too high),
        goto  WaitForLoopback   ;   clear all sample registers and start
                          ;   waiting all over again
        movlw LO_SUPPLY#  ; else set VAL corresponding to V=25.8V
        movwf VAL         ;
        call  PresetForSupply   ;
        call  DoCompCheck ;
        movwf TEMP1       ; Retrieve answer (0x00=yes, 0xFF=no)
        btfsc TEMP1,7           ; If no (Supply Voltage too low),
        goto  WaitForLoopback   ;   clear all sample registers and start
                          ;   waiting all over again
        movlw HI_IZ#      ; else set VAL corresponding to IZ = 32.8mA
        movwf VAL         ;
        call  PresetForSupply   ;
        call  DoCompCheck ;
        movwf TEMP1       ; Retrieve answer (0x00=yes, 0xFF=no)
        btfss TEMP1,7           ; If yes (Shunt Current too high),
        goto  WaitForLoopback   ;   clear all sample registers and start
                          ;   waiting all over again
        movlw LO_IZ#      ; else set VAL corresponding to IZ = 2.1mA
        movwf VAL         ;
        call  PresetForSupply   ;
        call  DoCompCheck ;
        movwf TEMP1       ; Retrieve answer (0x00=yes, 0xFF=no)
        btfsc TEMP1,7           ; If yes (Shunt Current too low),
        goto  WaitForLoopback   ;   clear all sample registers and start
                          ;   waiting all over again
        goto  ACK_now           ; else generate an ACK_tone

PresetForSupply
        clrf  TEMP1       ; Clear TEMP1
        bcf   TEMP1,CIS   ; Clear CIS to select Supply Voltage comparator
        bsf   TEMP1,VCOMP_OUT   ; Set VCOMP_OUT to indicate Supply Voltage
                          ; comparator output
        return                  ; return from PresetForSupply

PresetForIZ
        clrf  TEMP1       ; Clear TEMP1
        bsf   TEMP1,CIS   ; Set CIS to select Shunt Current comparator
        bsf   TEMP1,IZCOMP_OUT  ; Set IZCOMP_OUT to indicate Shunt Current
                          ; comparator output
        return                  ; return from PresetForIZ

PresetForTemp
        clrf  TEMP1       ; Clear TEMP1
        bsf   TEMP1,CIS   ; Set CIS to select Temperature comparator
        bsf   TEMP1,TEMPCOMP_OUT      ; Set TEMPCOMP_OUT to indicate Temperature
                          ; comparator output
        return                  ; return from PresetForTemp

DoCompCheck
        call  IsComp_gt_fVal    ; check whether the comparator input exceeds
                          ; f(Val) = 5*Val<3:0>/24 if Val<5>=1
                          ;          5*(0.25+Val<3:0>/32) if Val<5>=0
```

```
      return                    ; return from DoCompCheck (with unchanged W
register)

EvaluateAnswer
      movwf TEMP1         ; Put return value in TEMP1 (0x00 = yes, 0xFF = no)
      btfsc TEMP1,7             ; If no then
      goto  WaitForLoopback     ;   clear all sample registers and start
                          ;   waiting all over again
;     goto  ACK_now             ; else generate an ACK tone
                          ; (commented out due to sequentiality)

ACK_now
      movf  REP_ID,0      ; Select the tone...
      movwf TONE_ID             ; ...according to REP_ID
      goto  StartExtTimer       ; Prepare to generate a tone

IsComp_gt_fVal                  ; Is the comparator input greater than f(Val)?
      movlw CMODE#              ; Prepare for CM<2:0>=010
      iorwf TEMP1,1             ; Include CM<2:0> in TEMP1 (which has been preset
                          ;   according to the selected comparator)
      movf  TEMP1,0             ; Put TEMP1 in the W register
      movwf CMCON         ;   and then into the CMCON register
      movlw COUTMASK#     ; Mask the Comparator selection bits
      andwf TEMP1,1             ;   of TEMP1;  now it contains only a
                          ;   "1" in the selected comparator output
                          ;   position
      movlw VRMASK#             ; Mask the valid bits of VAL
      andwf VAL,0         ; and put result in the W register
      bsf   STATUS,RP0    ; Select bank 1
      movwf VRCON         ; Put valid bits of VAL in VRCON
      bsf   VRCON,VREN    ; Enable VREF
      bcf   STATUS,RP0    ; Select bank 0
      movlw .15           ;
      call  PauseN              ; Pause for a bit over 10us
      movf  CMCON,0             ; Get CMCON including comparator outputs
      andwf TEMP1,1             ; Clear all bits except the selected comparator
                          ;   output
      call  DefaultComparators      ; Restore comparators and voltage reference
                          ; to default state
      incf  TEMP1,1             ; Increment TEMP1
      decfsz      TEMP1,1           ; If the comparator output was high,
      retlw 0xFF          ;   answer = no
      retlw 0x00          ; else answer = yes

DefaultComparators
      bsf   STATUS,RP0    ; Select register bank 1
      movlw 0x88          ; Configure for internal voltage reference of 2.5V nom.
      movwf VRCON         ;
      bcf   STATUS,RP0    ; Select register bank 0
      movlw CMODE#              ; Set CMCON for
      movwf CMCON         ;   four independent comparators
      bcf   CMCON,CIS     ; Select Signal and Supply Voltage comparators
      movlw .15           ; Pause for
      call  PauseN              ;   a bit more than 10us
;*******************************************************************************
*****
      movlw 0x07          ; DEBUG - disable comparators, set for digital inputs
```

```
      movwf CMCON       ;
;*****************************************************************************
*****
      return                  ; return from DefaultComparators

PauseN
      movwf TEMP2       ; Put W register in TEMP2
PauseCycle
      decfsz      TEMP2,1           ; If not done counting down,
      goto  PauseCycle  ;   keep counting down
      return                  ; else return

ConcatenateBits
                        ; Shift TEMP1<7,6> into TEMP2<1,0>
      rlf   TEMP1,1           ; Rotate TEMP1 left through carry
      rlf   TEMP2,1           ; Rotate TEMP2 left through carry
      rlf   TEMP1,1           ; Rotate TEMP1 left through carry
      rlf   TEMP2,1           ; Rotate TEMP2 left through carry
                        ; Shift TEMP1 left twice
      rlf   TEMP1,1           ; Rotate TEMP1 left through carry
      rlf   TEMP1,1           ; Rotate TEMP1 left through carry
                        ; Shift TEMP1<3,2> into TEMP2<1,0>
      rlf   TEMP1,1           ; Rotate TEMP1 left through carry
      rlf   TEMP2,1           ; Rotate TEMP2 left through carry
      rlf   TEMP1,1           ; Rotate TEMP1 left through carry
      rlf   TEMP2,1           ; Rotate TEMP2 left through carry
      return                  ; return from ConcatenateBits

CheckParity
      movwf TEMP1       ; Move W register into TEMP1
      rrf   TEMP1,1           ; Rotate right two bits
      rrf   TEMP1,1           ;   through carry
      xorwf TEMP1,1           ; XOR with original TEMP1
      btfss TEMP1,0           ; If bit 0 parity is incorrect
      retlw .255        ; Return value .255
      btfss TEMP1,1           ; If bit 1 parity is incorrect
      retlw .255        ; Return value .255
      btfss TEMP1,4           ; If bit 4 parity is incorrect
      retlw .255        ; Return value .255
      btfss TEMP1,5           ; If bit 5 parity is incorrect
      retlw .255        ; Return value .255
      retlw .0          ; Return value .0

GetSamples
      clrf  COUNT1            ; set COUNT1=0
      movlw NSAMPLES#   ; initialize sample counter COUNT to NSAMPLES#
      movwf COUNT       ;
StartSampling
;*****************************************************************************
***
      ;btfsc      CMCON,SIGCOMP_OUT ; if Signal Comparator = 1 (high amplitude)
      btfsc PORTA,RA0   ; DEBUG - use RA0 instead
;*****************************************************************************
***
      incf  COUNT1,1    ; Increment COUNT1
      decfsz      COUNT,1           ; Decrement COUNT
                        ; PLACEHOLDER for additional delay between samples
```

```
        goto  StartSampling     ;
        movf  COUNT1,0      ; Put COUNT1 in W register
        movwf COUNT         ; Re-use COUNT to hold COUNT1 value
        movlw NMATCHING#    ;
        subwf COUNT,1             ; Subtract NMATCHING# from COUNT
        btfss COUNT,7             ; If remainder is positive,
        retlw 1             ;   then sample is valid, return with value "1"
        movf  COUNT1,0      ; Put COUNT1 in W register
        movwf COUNT         ; Re-use COUNT to hold COUNT1 value
        movlw NMATCHING#    ;
        addwf COUNT,1             ; Add NMATCHING# to COUNT
        movlw NSAMPLES#     ;
        subwf COUNT,1             ; Subtract NSAMPLES# from COUNT;
                            ; COUNT = COUNT1 - (NSAMPLES# - NMATCHING#)
        btfss COUNT,7             ; If remainder is positive,
        retlw .255          ;   then sample and message are invalid,
                            ;   return with value "0xFF"
        retlw 0             ; else sample is valid, return with value "0"

WaitForLoopback
        movlw 0xA0          ; Enable only the TMR0 overflow interrupt (T0IF),
                            ; set Global interrupt enable since ISR may
                            ; have been escaped without return, and clear all
                            ; interrupt flags
        movwf INTCON              ;
        movlw INIT_TMSB#    ; Initialize TIME_MSB register
        movwf TIME_MSB      ;
        clrf  PHASE         ; Set PHASE=0 (A)
        call  ClrSampleRegs       ; Clear all Phase A sample registers
        comf  PHASE,1             ; Set PHASE=0xFF (B)
ResetCurrentPhase
        call  ClrSampleRegs       ; Clear all sample registers of the current Phase
InfiniteLoop
        goto  InfiniteLoop        ; Wait!

ClrSampleRegs
        movlw ADDRN_A#      ; Starting with latest Phase A register
        btfsc PHASE,0             ; If PHASE = 1 (B)
        movlw ADDRN_B#      ;   switch to latest Phase B register
        movwf FSR           ; Initialize FSR to latest sample register
        movlw NUM_ADDRS#    ; Put number of sample addresses per phase
        movwf TEMP1         ;   into the TEMP1 register
ClrNextReg
        clrf  INDF          ; Clear the current sample register
        decf  FSR,1         ; Move to next earliest sample register
        decfsz      TEMP1,1             ; If not all registers are done,
        goto  ClrNextReg    ;   then go back to clear next register
        return                    ; return from ClrSampleRegs

SlowRampPORTB
        movlw .100          ; Pause about 72.5us
        call  PauseN              ;
        movf  PORTB,0             ; Read PORT_B
        movwf TEMP1         ;   into TEMP1
IncPORTB
        movf  TEMP1,0             ; Write TEMP1
        movwf PORTB         ;   into PORT_B
```

```
        movlw .100          ; Pause about 72.5us
        call  PauseN            ;
        incfsz      TEMP1,1             ; Increment TEMP1
        goto  IncPORTB      ; Prepare for another increment
                            ; (PORT_B is now .255)
        return                  ; return from SlowRampPORTB

StartExtTimer
        movlw 0x90          ; Enable only external INT interrupt
        movwf INTCON            ;
        call  SlowRampPORTB     ; Slowly ramp PORT_B back to .255
        bcf   PORTA,RA4     ; Trigger external timer via RA4
        bsf   PORTA,RA4     ;
        bcf   PORTA,RA4     ;
;       goto SelectTone         ; commented out due to sequentiality

SelectTone
        btfsc TONE_ID,RB1 ; If TONE_ID<1>=1,
        goto  tone15            ;   generate tone15
;       goto  tone10            ; else generate tone10 (commented out due to
sequentiality)

tone10                          ; 43.125 kHz sine at 2/96 UI intervals;  2
instructions overhead
        movlw   .253            ; 3
        movwf PORTB         ;
        movlw   .248            ; 5
        movwf PORTB         ;
        movlw   .242            ; 7
        movwf PORTB         ;
        movlw   .234            ; 9
        movwf PORTB         ;
        movlw   .224            ; 11
        movwf PORTB         ;
        movlw   .212            ; 13
        movwf PORTB         ;
        movlw   .198            ; 15
        movwf PORTB         ;
        movlw   .184            ; 17
        movwf PORTB         ;
        movlw   .169            ; 19
        movwf PORTB         ;
        movlw   .152            ; 21
        movwf PORTB         ;
        movlw   .136            ; 23
        movwf PORTB         ;
        movlw   .119            ; 25
        movwf PORTB         ;
        movlw   .103            ; 27
        movwf PORTB         ;
        movlw    .86        ; 29
        movwf PORTB         ;
        movlw    .71        ; 31
        movwf PORTB         ;
        movlw    .57        ; 33
        movwf PORTB         ;
        movlw    .43        ; 35
```

```
movwf PORTB         ;
movlw   .31         ; 37
movwf PORTB         ;
movlw   .21         ; 39
movwf PORTB         ;
movlw .13           ; 41
movwf PORTB         ;
movlw   .7          ; 43
movwf PORTB         ;
movlw   .2          ; 45
movwf PORTB         ;
movlw   .0          ; 47
movwf PORTB         ;
movlw   .0          ; 49
movwf PORTB         ;
movlw .2            ; 51
movwf PORTB         ;
movlw .7            ; 53
movwf PORTB         ;
movlw .13           ; 55
movwf PORTB         ;
movlw .21           ; 57
movwf PORTB         ;
movlw .31           ; 59
movwf PORTB         ;
movlw .43           ; 61
movwf PORTB         ;
movlw .57           ; 63
movwf PORTB         ;
movlw .71           ; 65
movwf PORTB         ;
movlw .86           ; 67
movwf PORTB         ;
movlw .103          ; 69
movwf PORTB         ;
movlw .119          ; 71
movwf PORTB         ;
movlw .136          ; 73
movwf PORTB         ;
movlw .152          ; 75
movwf PORTB         ;
movlw .169          ; 77
movwf PORTB         ;
movlw .184          ; 79
movwf PORTB         ;
movlw .198          ; 81
movwf PORTB         ;
movlw .212          ; 83
movwf PORTB         ;
movlw .224          ; 85
movwf PORTB         ;
movlw .234          ; 87
movwf PORTB         ;
movlw .242          ; 89
movwf PORTB         ;
movlw .248          ; 91
movwf PORTB         ;
```

```
        movlw .253          ; 93
        movwf PORTB         ;
        movlw .255          ; 95
        movwf PORTB         ;

        goto  tone10              ; Repeat until interrupt

tone15                            ; 64.6875 kHz sine at 2/64 UI intervals;  2
instructions overhead
        movlw   .250              ; 3
        movwf PORTB         ;
        movlw   .240              ; 5
        movwf PORTB         ;
        movlw   .227              ; 7
        movwf PORTB         ;
        movlw   .209              ; 9
        movwf PORTB         ;
        movlw   .188              ; 11
        movwf PORTB         ;
        movlw   .165              ; 13
        movwf PORTB         ;
        movlw   .140              ; 15
        movwf PORTB         ;
        movlw   .115              ; 17
        movwf PORTB         ;
        movlw   .90         ; 19
        movwf PORTB         ;
        movlw   .67         ; 21
        movwf PORTB         ;
        movlw   .46         ; 23
        movwf PORTB         ;
        movlw   .28         ; 25
        movwf PORTB         ;
        movlw   .15         ; 27
        movwf PORTB         ;
        movlw   .5          ; 29
        movwf PORTB         ;
        movlw   .0          ; 31
        movwf PORTB         ;
        movlw   .0          ; 33
        movwf PORTB         ;
        movlw   .5          ; 35
        movwf PORTB         ;
        movlw   .15         ; 37
        movwf PORTB         ;
        movlw   .28         ; 39
        movwf PORTB         ;
        movlw .46           ; 41
        movwf PORTB         ;
        movlw   .67         ; 43
        movwf PORTB         ;
        movlw   .90         ; 45
        movwf PORTB         ;
        movlw   .115              ; 47
        movwf PORTB         ;
        movlw   .140              ; 49
        movwf PORTB         ;
```

```
movlw .165          ; 51
movwf PORTB         ;
movlw .188          ; 53
movwf PORTB         ;
movlw .209          ; 55
movwf PORTB         ;
movlw .227          ; 57
movwf PORTB         ;
movlw .240          ; 59
movwf PORTB         ;
movlw .250          ; 61
movwf PORTB         ;
movlw .255          ; 63
movwf PORTB         ;

goto  tone15                ; Repeat until interrupt

   END              ; That's all Folks !
```